An Artificial Immunology Inspired Approach to Achieving Self-Expression in Collective Adaptive Systems

NICOLA CAPODIECI, University of Modena and Reggio Emilia
EMMA HART, Edinburgh Napier University.
GIACOMO CABRI, University of Modena and Reggio Emilia

Distributed autonomous systems consisting of large numbers of components with no central control point need to be able to dynamically adapt their control mechanisms during run-time in order to deal with an unpredictable and changing environment. Existing frameworks for engineering self-adaptive systems fail to account for the need to incorporate *self-expression*, i.e. the capability of a system to dynamically adapt its coordination pattern during run-time. Although the benefits of incorporating self-expression are well-known, there is currently no principled means of enabling this during system design. We propose a conceptual framework for principled design of systems that exhibit self-expression, based on inspiration from the natural immune system. The framework is described as a set of design principles and customisable algorithms, and then instantiated in three case-studies, including two from robotics and one from artificial chemistry. We show that it enables self-expression in each case, resulting in systems that are able to adapt their choice of coordination pattern during run time to optimise functional and non-functional goals as well as to discover novel patterns and architectures.

Categories and Subject Descriptors: D.2.10 [**Software Engineering**]: Design; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence; F.1.11 [**Computation by Abstract Devices**]: Models of Computation

General Terms: Autonomic Computing, Artificial Immune System, Framework

ACM Reference Format:
ACM Trans. Autonom. Adapt. Syst. V, N, Article A (January YYYY), 25 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Systems consisting of ensembles of software components (e.g. devices, robots, sensors) are now ubiquitous. Typically, these systems consist of large numbers of possibly heterogeneous components, operating without any central point of control in potentially unknown or dynamically changing environments. In order to achieve collective and/or individual goals in such environments, the ensemble must be able to restructure its topology or control regime during run-time as appropriate. This ability to enable run-time selection of architectural and coordination patterns in response to environmental changes was coined *self-expression* by [Zambonelli et al. 2011]. It proposes that ensembles should be able to deploy just-in-time reconfiguration of topologies in terms of interactions, roles and behaviours of the components when required. This might lead to switching from a hierarchy to a collective of peers for example, or switching from being a collective decision-making ensemble to a competitive market-based one.

[Cabri and Capodieci 2013] describe a number of diverse case-studies that could benefit from self-expression by changing run-time behaviours, including space-exploration, self-organising smart-grids and autonomous robotics. [Zambonelli et al. 2011] further discusses its application in autonomic service-component ensembles. A number of approaches to enabling self-expression have been proposed. [Cabri et al. 2014] consider how to enable self-expression by exploiting a formal language for defining ensembles; [Puviani et al. 2014] discusses how it can be achieved through the use of *roles*; [Capodieci et al. 2014] propose an approach that takes inspiration from holonic organisation in multi-agent systems. However, the proposed approaches currently lack any common elements, making it difficult for a system designer to methodically approach the design of a new system. In this article, we address this issue

by describing a novel conceptual framework for designing and building autonomous systems that exhibit self-expression. Our approach takes inspiration from the natural immune system, and in particular a theory described by Cohen called the Cognitive Network Theory [Cohen 2000b]. Cohen hypothesis that the immune system is able to make collective decisions based on the simultaneous sensing of environmental signals by multiple cells: this enables a coordinated response to emerge that adapts as the environment changes, a process exactly akin to the idea self-expression proposed by [Zambonelli et al. 2011]. We describe a conceptual framework that consists of a set of design principles and generic algorithms (described by pseudo-code) that can be customised to specific applications in order to achieve self-expression. The framework is instantiated in three separate case studies:

— A study in *Swarm Robotics* shows that self-expression leads to adaptation of the swarm to different environments, with mixed coordination patterns occurring depending on specific features of the environment
— An additional robotics study that examines an ensemble of inter-connected components that aggregate to form a virtual creature that has to optimise its behaviour with respect to both functional and non-functional requirements
— A study from the *morpho-genetic engineering* domain which evaluates the ability of the proposed approach to find, generate and detect novel emergent architectures in ensembles of self-propelled agents.

In each case, the basic algorithms provided by the framework are shown to be easily customised to the specific applications. We show that using the framework, it is possible to build systems that are capable of selecting an optimal coordination pattern or organisational architecture during run-time, in environments that are task-driven and may additionally have non-functional requirements. The ability to perform run-time selection enables systems to adapt to a range of environments without any explicit human intervention. Given its generic nature, we propose that the framework can be adapted to a wide and diverse range of applications in domains such as multi-agent systems, artificial life and autonomic computing amongst others.

The paper is organized as follows: in Section 2 a background review of self-expression and the immune paradigm that inspires this work. The main description of the framework is then discussed in Section 3, while in Section 4, shows the principles embodied in the framework are instantiated in three different case studies. Discussions about the proposed framework and the lessons learned in its applications will conclude the paper in Section 5.

2. RELATED WORK AND BACKGROUND

In this section we introduce the notion of self-expression and briefly review current approaches to building distributed systems that exhibit self-expression. We then introduce the AIS paradigm, paying particular attention to the notion of Immune Networks that inspire this work.

2.1. Self-Expression

Self-expression can be considered as an additional *self-** property that ensembles of autonomic components need to deploy. A given ensemble can potentially solve a problem in multiple ways, each of which requires different collaborative behaviours between the agents composing the ensemble. A *coordination pattern* describes the mechanism via which components collaborate. For example, in the robotics domain, [Fernandez-Marquez et al. 2013] describe several patterns for movement, including flocking, foraging and chemotaxis. Other patterns are proposed by [De Wolf and Holvoet 2007] for building autonomic systems, that include gradient fields, pheromone paths, and

market based coordination. The coordination pattern selected will determine the extent to which the functional and non-functional requirements of the system are met, with the choice of appropriate pattern often dependent on environmental conditions. Self-expression is therefore the ability of an ensemble of autonomic components inserted in an open and non-deterministic environment to change its coordination pattern during the *run-time* execution of a task: it deals with just in time reconfigurations of topologies of interactions, roles and behaviours of the components in a distributed architecture and manifest itself whenever the variation in the external environment threatens the stability of the system. The benefit of a system inserted in an open and non-deterministic environment being able to change its coordination pattern during the *run-time* execution of a task has been clearly outlined in previous research [Cabri and Capodieci 2013]. Further experimental evidence is provided in [Puviani et al. 2013].

A recent roadmap [De Lemos et al. 2013] discussed open challenges for the community in engineering self-adaptive systems, highlighting that the dynamic nature of self-adaptation hinders the applicability of traditional software engineering principles and practices. In particular, [De Lemos et al. 2013] highlight the need for an approach that includes issues such as how to formally express the *observation* of the surrounding environment; how to correctly *represent* tasks, components and goals; how to *control* the decision making process so to be able to *identify* possible solutions; how to *enact adaptation*, i.e. selection of the *fittest* solution. We propose that the Artificial Immune System paradigm embodies exactly the properties just outlined.

2.2. AIS

Artificial Immune Systems (AIS) represent a computational approach for designing intelligent systems and covers a variety of algorithms that have been used in a multitude of different applications [Hart and Timmis 2008]. As a metaphor, it has particular appeal to the system engineer attempting to design self-* systems: it is decentralized, scalable, performs self/non self discrimination and learning, resulting in a self-configuring, self-protecting, self-healing and self-optimising system. It also demonstrates self-expression, i.e. the ability to select between a number of responses depending on the situation. At a system level, this ranges from choosing between simple physiological changes such as increasing body temperature, through deploying innate responses (non-adaptive genetically coded immune responses) to initiating a response from the *adaptive* immune system. The latter sub-system is of particular in interest in both computer science and engineering, in endowing a system with the ability to learn new responses to previously unseen scenarios. Although a number of different mechanisms have been proposed to explain immune functionality, of most relevance to this discussion is the immune network (or idiotypic network) theory first proposed by Jerne in [Jerne 1974] and subsequently recognised in the engineering literature as a decentralised, consensus-making system.

A simplified view of the immunological detail is sufficient for our purpose: assume the immune system is composed of a set of diverse immune cells known as *antibodies* whose function is to detect pathogenic material (*antigens*). Jerne postulated that antibodies are capable not only of binding to pathogenic material but also to other antibodies, creating a dynamic network in which elements of the network either suppress or stimulate each other depending on the mode of binding. Interactions with the environment and other immune cells regulate the concentration of antibodies, dynamically altering the topology and composition of the network. The network model is proposed to facilitate both cognition and memory [Cohen 2000a; 2000b], in that the suppression and stimulation signals that regulate concentration lead to decision making by pro-

moting the most relevant (highly concentrated) antibody at any given time (cognition) and sustaining previously useful antibodies over long periods (memory).

According to this model, a network of antibodies exists that continuously senses the environment. Depending on its specific receptors, each antibody binds to a different degree to signals sensed from external observable environment (dubbed *situation-oriented antigens* by [Ishiguro et al. 1995]) and to *goal-related antigens* that indicate how well it performs on the task under consideration. The strength of the binding regulates the concentration of the antibody, with strong binding resulting in an increase in concentration and vice versa. In addition, each antibody interacts with other antibodies in the network. Again depending on specific receptors, these interactions may be stimulatory, increasing the concentration of an antibody, or suppressive, decreasing its concentration. In essence, the interactions between antibodies define preference relationships between antibodies, given the current environmental conditions and past performance of the antibody. The concentration level of an antibody thus reflects its past history, indicating how often it has been selected in the recent past, and its relative performance. Action results from the network through selection of the antibody with highest concentration. Differential equation models that describe the self-regulation of concentration accounting for all factors just outlined were proposed in the immunological literature [Jerne 1974]. They have subsequently been adopted and adapted in engineered systems, particularly to implment dynamic behaviour arbitration mechanisms in robotic applications, e.g. [Ishiguro et al. 1995]. A diagram summarising these interactions is shown in Figure 1.

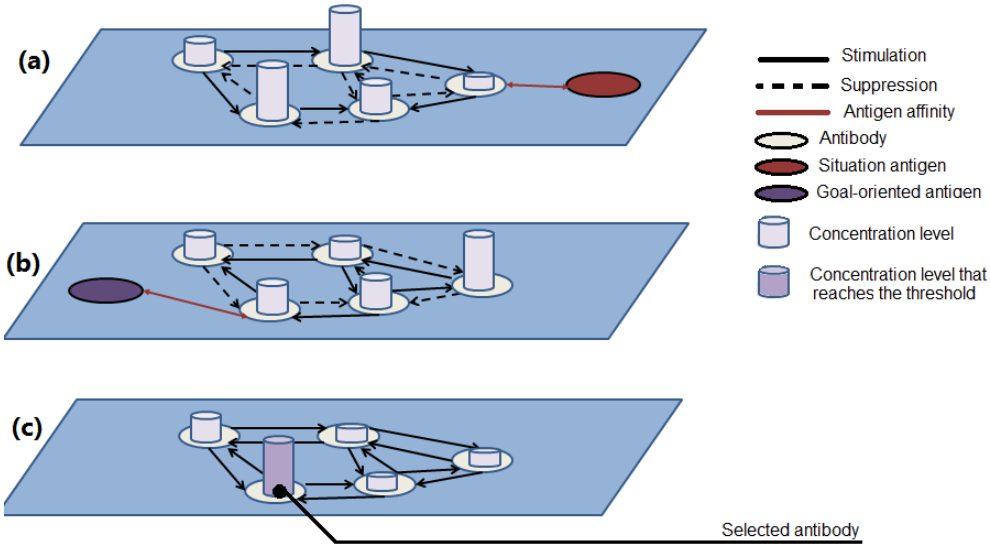

Fig. 1. Dynamics of an implemented idiotypic network. (a) stimulation and suppression signals vary the concentration of the antibodies according to the distance between an antibody and an antigen (perceived environmental conditions). (b), concentration is also affected by performance w.r.t. the goal oriented antigen. (c), concentrations stabilize; an action is selected according to the antibody with highest concentration.

In summary, it is apparent that the paradigm has a natural mapping to the key elements of the *Design Space* proposed in [De Lemos et al. 2013]: it provides a mechanism to express the *observation* of the surrounding environment (pathogens, signals); there

are structures for *representing* tasks, components and goals (antibodies); it offers a solution to the *control* issue through a decision making process (varying concentration through self-regulation of network) so to be able to *identify* possible means to *enact adaptation*, i.e. selection of the *fittest* solution (selection of highest concentrated antibody). In the next section, we present a conceptual framework that maps these ideas into a conceptual framework. The framework consists of a set of design principles and generic algorithms that can be customised to achieve self-expression in a generic distributed system. We assume the distributed system itself consists of a number of decentralised components, each of which exhibits run-time selection of an appropriate coordination pattern in response to its perception of the current environment.

3. THE CONCEPTUAL FRAMEWORK

An overview of the proposed framework is given in Figure 2.

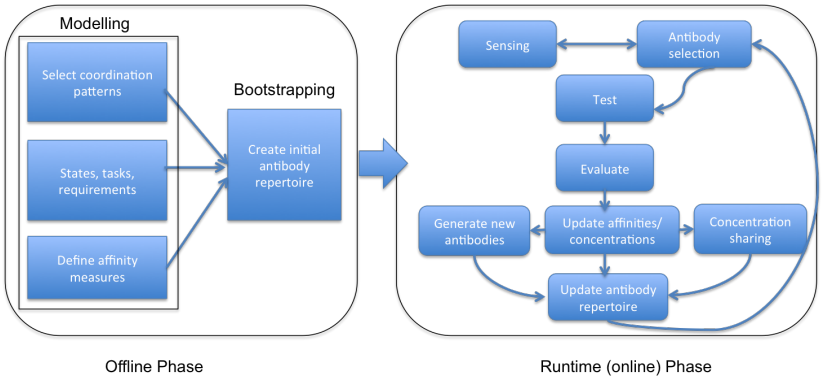

Fig. 2. An schematic overview: the left hand diagram shows the offline phase (detailed in Section 3.1.2). The right hand diagram represents the runtime adaptation mechanisms (detailed in Section 3.2)

The key element of the framework is an *antibody* that defines an environmental state, the preferred coordination pattern to select in response and the expectation of the utility gained from applying the pattern. Antibodies are connected in a network with a dynamically changing topology inside a lymph-node that resides within each component. The artificial idiotypic network is regulated through two separate processes:

— a *bootstrapping phase* whose purpose is to create an initial knowledge base consisting of appropriate responses to differing environmental conditions. Such phase is executed off-line.
— a *run-time phase* which determines selection of appropriate coordination pattern through modelling dynamics in concentration of antibodies within lymph-nodes; also refines knowledge base over time, proposing new antibodies and removing redundant ones. This phase is executed periodically at run-time.

After detailing the manner in which we model the problem, each phase is discussed in detail below.

3.1. Modelling

We assume a computational problem that involves collaboration among a set of components E in order to solve a single task τ. The task is global in that all components

should contribute to solving the same task and we assume that the task does not change during runtime. Given this, we define the following:

— A component $C_x \in E$, described as a tuple $< L_{c_x}, sensors, actuators >$, in which L_{c_x} refers to an associated *lymph-node*
— A lymph-node L_{c_x} is associated with each component and contains a set of interconnected *antibodies* A, acting as an evolving knowledge repertoire
— An antibody is represented as a tuple $< Conditions,\ Action,\ Expectations >$
— An affinity metric $r_{i,j}$ represents the affinity value of the antibody i towards antibody j
— A utility metric U that defines the utility of the ensemble; this can consist of purely functional requirements or additionally include non-functional requirements.
— E is responsible for solving a computational problem defined as a *task* τ
— A task τ is represented as a tuple $(fReq, nfReq)$ defining a set of functional and non-functional requirements to be optimised.

The *antibody* is the key data structure of the system. The first field of the antibody tuple, $Conditions$, represents an antibody's expectation of the current environmental conditions. This can be directly compared to the actual environmental conditions perceived through the component's sensors through the use of an *affinity metric*. This metric specifies a distance between two condition vectors, therefore its definition is dependent on the specific application and data collected. The $Expectations$ field denotes the expected utility U of applying the single action specified in the $Action$ field. Expectations can be defined in terms of both functional and non-functional requirements — again these are application specific. A distance measure is also required to determine the distance between the expected utility as defined in an antibody and the actual utility as recorded from the application. This is also application specific, dependent on the types of data involved. A detailed discussion of distance metrics appropriate to a range of type of data is given in [Freitas and Timmis 2007]. The remaining $Action(s)$ field within the tuple field deals with the coordination pattern to be applied and is described below.

A task requires optimisation of one or more functional requirements and optionally, a set of non-functional requirements, from an initial starting state. The task specification is known to all the components. In the case studies reported, the task remains fixed throughout each experiment. Note that if required, the task can be formally defined, e.g. using the *State Of The Affairs* (SOTA, [Abeywickrama et al. 2012]) methodology for describing states, requirements and tasks in combination with the *Software Component Ensemble Language* (SCEL, [De Nicola et al. 2014]), as seen in [Bures et al. 2013]. In another paper [De Nicola et al. 2014] we conducted a study in which we consider systems in which run time changes in task specification or objectives can occur. However this latter paper is oriented towards formal modelling and task representation, rather than describing the immune-inspired mechanism of adaptation of relevance here.

3.1.1. Coordination patterns. For a given application, a set of appropriate coordination patterns must be defined. A coordination pattern is characterized by a set of roles and interactions, which typically use different subsets of the sensors and actuators of each component. Typically, different coordination patterns designed to achieve the same task will show commonalities at the level of code and process computation [Cabri and Capodieci 2013], but are likely to solve the task with varying utilities and performance.

During runtime, each component in the ensemble selects a *single* coordination pattern to deploy. Within the selected pattern, the component will adopt *one* of the roles defined by the pattern — roles are pre-defined within each pattern, and in general

are selected probabilistically by a component. Thus, during runtime, different components may execute different patterns. From a global perspective, this may result in the swarm deploying a novel coordination pattern that is a composite of the original patterns. Additionally, for a given subset deploying the same pattern, different components may execute different roles. The specific pattern to be used by a component is defined in the central field of the antibody tuple by the $Action(s)$.

3.1.2. Generation of initial repertoire. A one-off bootstrapping phase is carried out by the ensemble in order to construct an initial repertoire of antibodies, i.e. define the condition, action and expectation fields of the antibodies. This requires the task τ to be executed using each of the coordination patterns individually; for each pattern, the task is repeated under a range of initial starting conditions. For each starting condition $< Conditions >$, an antibody records the coordination pattern tested in the $< Action >$ field and the utility U achieved in the $Expectation$ field. A simulation tool is a powerful aid in this step in ensuring that many configurations can be tested.

At the end of each test, a knowledge base is created consisting of antibodies defined by an initial condition (condition field), a single action, and the expected utility. A default level of concentration and inter-antibody affinity is assigned to each antibody and the newly created antibodies are assigned to lymph-nodes. Two options are apparent: (i) assign the same set of antibodies to every lymph-node (low diversity) or (ii) create lymph-nodes with unique repertoires (high diversity). Although the latter approach is common in fields such as evolutionary algorithms in order to aid exploration of the search space, we propose that this is not alway necessary in this model: in typical distributed applications, components of large ensembles that share identical repertoires are likely to be subject to widely differing experiences, hence causing differentiation within the repertoires and therefore the necessary diversity required for exploration.

3.2. Runtime mechanisms

During runtime, the ensemble is governed by a process that has three crucial roles:

Selection and Evaluation. For each component, select the most appropriate action to apply at time t. Action selection depends on antibody *concentration* within a specific lymph-node: each antibody has a dynamically changing concentration level c that depends how well its $condition$ variable matches the environment, the frequency and recency with which it has been selected, and feedback it receives from other antibodies in the network after applying its action. The antibody in a lymph-node that has highest concentration applies its action. The designated action is applied for an evaluation time T_{eval}, so that its utility U_a (according to functional and non-functional requirements) can be measured. This is then compared to the expected utility E_a in order to determine whether the antibody should receive positive feedback (due to outperforming its expected utility) or negative feedback in the case it under performed. More detail on how the concentration update for the selected antibody is operated is given in Section 3.2.1.

Generation. This process generates novel antibodies (i.e in addition to those discovered in the bootstrapping phase) within a lymphnode through applying mutation operators to the antibodies. Such process is detailed in Section 3.2.2.

Sharing. This process governs the sharing of information *between* components, enabling feedback loops between components: over time, the antibody composition and concentration levels within a lymph-node reflect the history of experiences within that component. Sharing this information with neighbouring components enables useful information to be spread throughout the distributed system. This is described in more detail in Section 3.2.3.

Pseudo-code is given in Algorithm 1 and described in detail in the following sections.

Algorithm 1 Runtime

```
1: Initialise
2: repeat
3:     for <each lymph node l> do
4:         a* ← antibody with maximum value c_a
5:         Apply action from a* for t time steps
6:         U_a* ← utility after applying action from a*
7:         if (U_a* > E_a*) then
8:             feedback to a* → positive
9:         else
10:            feedback to a* → negative
11:        end if
12:        for <each antibody node a> do
13:            C_d ← distance of condition field to current environment
14:            Nf ← non-functional requirements
15:            c_a ← update antibody concentration according to equation 1(a).
16:        end for
17:        Generate new antibodies via mutation
18:        Share concentration with other lymph-nodes
19:    end for
20: until stopping criteria met
```

3.2.1. Concentration Update. The value of the concentration of each antibody over time depends on three factors: its stimulation or suppression from other antibodies in the lymph-node, its distance to the current environmental state and the state if its non-functional requirements, and a decay factor that diminishes its concentration over time. Concentration is calculated via Eq. 1 which is borrowed directly from the original work by [Ishiguro et al. 1995] (in turn derived from the immunological theories of Jerne [Jerne 1974]) and have been shown to be effective in performing behaviour selection within a single mobile robot.

$$
\begin{aligned}
(a)\tfrac{\Delta c_i}{\Delta t} &= K_p \sum_{j=1}^{Na} r_{j,i} c_i c_j - K_n \sum_{k=1}^{Na} r_{i,k} c_i c_k + K_r f(nfReq, C_D) - K_d c_i \\
(b) r_{i,j} &= \frac{T_{ni} + T_{pj}}{T_{i,j}} f(fReq) \quad r_{j,i} = \frac{T_{nj} + T_{pi}}{T_{i,j}} f(fReq) \\
(c) c_i &= \frac{1}{1 + exp(0.5 - c_i)}
\end{aligned}
\tag{1}
$$

The difference equation (Eq. 1(a)) regulates the concentration over time ($\frac{\Delta c_i}{\Delta t}$ according to four separate terms, each of which is regulated by constants (K_p, K_n K_r and K_d). More specifically:

(1) The first term accounts for *stimulation signals* received by antibody i in the case that its calculated utility exceeds that given in its Expectation term, i.e. it represents antibody i receiving positive feedback from other antibodies (designated j) and increases the likelihood of selecting i in future

(2) The second term accounts for *suppression signals* received by antibody i in the case that its calculated utility is less than that given in its Expectation term, i.e. it represents antibody i receiving negative feedback from other antibodies (designated k) and decreases the likelihood of selecting i in future
(3) The third term is an application dependent function that depends on the extent to which the non-functional requirements ($nfReq$) have been met and on the distance (C_D) between the currently perceived condition from the environment and the condition field of the antibody
(4) The final term is a *decay rate* that represents the tendency of an antibody to die in absence of any stimulation.

The first and second terms are regulated by the *affinity* $r_{i,j}$ between two antibodies which is defined in Equation 1(b). The resulting balance between stimulation and suppression captured in these terms serves as an indicator of the relative preference of one antibody over another. An example is useful to clarify:

Assume $Ab1$ and $Ab2$ are stimulated by the same antigen (situation), and than $Ab2$ is randomly selected. Applying the action results in the antibody receiving either negative or positive feedback depending on the utility obtained. Now assume it obtained positive feedback: to ensure that the system will tend to select $Ab2$ over $Ab1$ in a subsequent iteration, we recalculate the affinities as shown in Eq. 1(b) which raises the relative preference of $Ab2$ over $Ab1$. T_{ni} and T_{pi} indicate the number of times in which the selection of antibody i resulted in negative or positive feedback respectively. $T_{i,j}$ is the total number of times that both these antibodies have been selected. The third term directly captures the relationship between the antibody condition field and the environment, but can also be influenced by non-functional requirements — this is strictly application dependent and therefore not always necessary. Finally, Eq. 1(c) represents a squashing function used in order to ensure stability in values of the calculated concentration.

After the new concentrations of each antibody have been determined, various methods can be applied to select a new antibody to apply its action. The simplest of these is to use deterministic selection of the highest concentrated antibody, but other methods such as fitness-proportionate selection could be applied to introduce an element of stochasticity if required (e.g. a fitness proportionate selection as seen in [Ishiguro et al. 1995]).

3.2.2. Generation of novel antibodies. New antibodies are generated at the *micro* level by applying a mutation operator to antibodies within a lymph-node. In the original work of [Ishiguro et al. 1995], mutation operators were applied to antibodies at each generation to perform maximum exploration of the search space. In contrast, in the algorithm described here, mutations are triggered by a stagnation condition that occurs whenever the maximum level of concentration observed within a lymphnode is exhibited by more than one antibody. An appropriate mutation operator must be designed — this needs to be tailored to the specific application, but could for example include applying random perturbations to selected values, or combining values from more than one antibody into a new one. An exhaustive survey of generic mutation and crossover operators is summarized in [Floreano and Mattiussi 2008].

Mutations at the micro-level as described are able to generate novel *coordination patterns* from a macro perspective: each unit/lymph-node of the considered system acts autonomously by taking decisions as to which coordination pattern to follow based on its own experiences. As subsets of the ensemble can express different coordination patterns at any given time, then from a global perspective, novel coordination patterns emerge when considering the swarm as a whole. Hence, the search space of possible *solutions* is expanded.

3.2.3. Sharing information across the ensemble. The concentration of an antibody over time represents the history of how well the specific antibody tackled the specified task. Antibodies residing within different lymph-nodes may develop different histories depending on their own experiences, thus it could be useful for a single component to take into account how other components within the same ensemble are performing. This potentially enables one component to adapt to other component's experiences through sharing of information regarding antibodies and concentration. Again, this is an application dependent detail and therefore is described individually for each case study. The key point is that the process puts in place a basis for distributed and decentralised feedback loops within the ensemble.

4. APPLICATIONS

In this section we illustrate the application of the framework in three different case studies, with the goal of demonstrating that the adaptive characteristics of idiotypic networks can be used to control a variety of computational systems. Rather than focussing on results, this section highlights the relevant design choices in terms of the framework and discusses the implications of these choices. As each study has been described in detail in previously published conference proceedings, the reader is referred to the corresponding publication [Capodieci et al. 2013; 2014b; 2014a] for a more detailed discussion of the results.

4.1. Swarm robotics

The first case study used as test case for the proposed design guidelines involves a simple foraging scenario in swarm robotics [Capodieci et al. 2013]. The purpose of the case study is to show that approach is able to adapt at runtime its choice of coordination pattern(s) to optimise task performance in different environments. The case-study consider functional requirements (task optimisation) only. Two environments are considered: the first is a simple rectangular space with no obstacles; the second arena uses an extensive sized hexagonal arena with an obstacle obstructing the direct path from nest to food area. The ARGoS simulator was used for our experiments [Pinciroli et al. 2012].

We first describe the task and the specifics of the simulated swarm of robots and their initial coordination patterns. Following these definitions, we then describe how the generic framework outlined above can be specified for this particular application.

4.1.1. Related Work on AIS and Swarm Robotics. A fruitful line of work within robotics that started with [Ishiguro et al. 1995] has applied inspiration from Jerne's idiotypic network theory to develop behaviour arbitration mechanisms in individual robots. Antibodies consist of a condition that matches environmental conditions, an action, and a set of receptors that enable interactions with other antibodies. The resulting network of stimulatory and suppressive connections alters concentrations of antibodies; the one with the highest concentration applies its action. Various weaknesses in this work that required hand-coding of antibodies for instance have recently been addressed in [Whitbrook et al. 2010b], who consider evolutionary methods for generating antibodies and reinforcement learning to connecting them in a network. This resulted in a system that has been ported successfully to real-robots [Whitbrook et al. 2010a]. We extend this work in that we deal with *swarms* of robots rather than individuals, and that rather than considering individual actions, the robots must select a cooperative strategy to take part in.

4.1.2. Task. A swarm of robots, initially randomly distributed in a confined space called the *arena*, are required to collect food from a source and return that food to a nest. The goal of the task is to maximise the total amount of food returned. The robot

used in the simulation is called a *footbot*, which has only simple sensors and actuators: a robot is unable to calculate its absolute position in space, nor does it have any concept of orientation and can only carry one unit of food at a time[1]. A graphical depiction with a schematic representation of how the antibody tuple is modelled is reported in Figure 3.

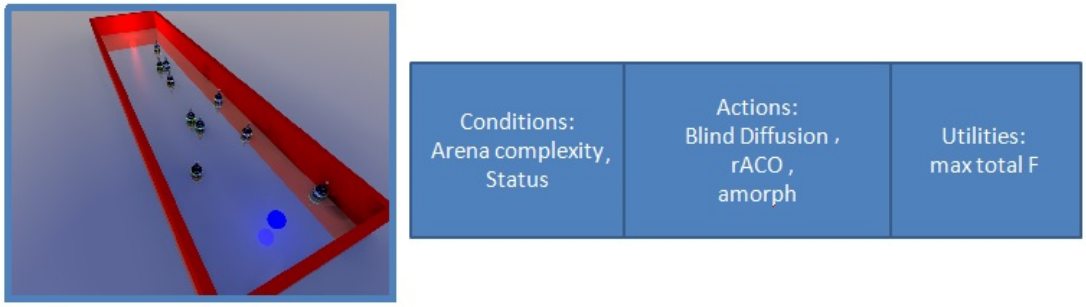

Fig. 3. (Left): a screenshot of the robot simulator (ARGoS [Pinciroli et al. 2012]): robots are initially randomly distributed in the arena, characterized by a nest area (blue orb) and a food area (red orb). (Right): a schematic representation of a modelled antibody (F = number of collected food units.

The task, in terms of initial and final state can be represented as starting from $\langle t = 0, \#FoodUnits = 0 \rangle$, the final state can be written as: $\langle t = L * T_{eval}, \#FoodUnits \rangle$, with $L \in \mathbb{N}$ is the total number of iteration of the on-line dynamics, such as multiplied by T_{eval} (time length of a single iteration) gives us the total duration of the experiment. $\#FoodUnits$ is the quantity to maximize (collected food units).

4.1.3. Coordination Patterns. Three potential coordination patterns are implemented and are available as the *actions* to be specified in the relevant field of an antibody: (i) a baseline and purely emergent behaviour, (ii) a swarm approach with only stigmergic communication and (iii) a peer-to-peer (p2p) approach with fixed communications. Each coordination pattern uses different subsets of sensors and actuators of the footbot robot, and within each pattern, different roles are defined.

A full description of all three coordination patterns can be found in [Capodieci et al. 2013]. We describe them briefy here. The first, *blind diffusion* is the most simple, containing only one role, and tends to lead to the swarm becoming uniformly distributed throughout the whole arena. This algorithm is likely to be inefficient with respect to the amount of food collected, but is useful in providing an estimation of the arena complexity (described later) and to periodically rearrange the robots positions. The second pattern used is *rACO*, which stands for robotic Ant Colony Optimization (ACO). This is a robotic application of the well-known ant colony optimization family of algorithms [Dorigo et al. 1996]. The pattern defines two roles: a proportion of the swarm act as *pheromones*, stationing themselves on paths that lead to food sources, and indicating distance to the source by altering the intensity of a light. The remaining *active robots* follow the pheromone trails to collect food[2]. The final pattern *amorh* is a *p2p* approach, inspired by [Abelson et al. 2000] which describes the use of bio-inspired

[1]For a more complete description of the robot used in the simulations, see page 3 http://iridia.ulb.ac.be/IridiaTrSeries/rev/IridiaTr2012-001r001.pdf

[2]A video illustrating the ant robots attracted by the trail of yellow lights is provided at https://vimeo.com/101705747

algorithms for achieving collaboration among a potentially large number of devices connected through unstructured topologies. The coordination pattern strongly relies on communication in the form of packets sent and received with the RAB (Range And Bearing) sensor and actuator built in each robot (i.e. infra-red communications). The pattern defines two specific roles, that of path-opener and path-closer, allocated to one robot each. All other robots diffuse according to signals received. An emergent property of this algorithm is that the path of lit robots formed is exactly one-robot wide[3].

As a general point, we note that when selecting coordination patterns, it is preferable that coordination patterns share common elements, thus following the *scaffolding* principle proposed in [Orosz 2001]. This reduces the probability that antagonistic behaviours might emerge through different subsets of the swarm selecting different patterns. Table I shows commonalities between the functions contained in each of the three coordination patterns described above; given that the patterns share a number of features, we propose that this limits the potential for conflicting patterns to emerge. However, we note there is still some onus on the designer to make intelligent choices when defining coordination patterns, for example a particular coloured light used in one pattern should have the same meaning in another pattern.

Table I. Common behaviours among the different coordination patterns used in the swarm-robotic case study.

	Diffuse	**rACO**	**AMORPH**
Diffuse Navigation	X	X	X
Load-unload food	X	X	X
Follow trail		X	X
Act as Pheromone		X	
Gradient distribution			X

4.1.4. Model and Bootstrapping phase. Trivially, the swarm is our Ensemble E of components with each robot being associated to a virtual lymphnode (see Figure 3). The initial repertoire of antibodies is created during the bootstrapping phase described below: at the end of this phase, each lymphnode/robot hosts an identical collection of antibodies.

The antibody (as shown in Figure 3, left side) is modelled as follows:

— Utility: maximize $\#FoodUnits$ during each time interval
— Actions: {Blind diffusion, rACO, amorph}
— Conditions:{complexity, status}

Complexity is a real-value between 0 and 1 representing the complexity of the arena, and is estimated in the bootstrapping phase as the number of times a robot can traverse the area in a fixed time interval. Thus, 0 indicates highest complexity. Status is either $\langle 0, 1, t \rangle$ where 0 indicates an ant robot, 1 a node robot if the amorph model is selected, and t indicates the time a robot has spent in the pheromone state if the rACO coordination pattern is currently selected. More specifically, the status condition is used to calculate the affinity changes among antibodies inside the same lymph-node as it is essential to account for the amount of time that specific lymph-node acted as a pheromone or a node instead of a collecting robot. More specifically, the affinity $r_{i,j}$ of a single antibody is calculated using a weighted parameter that takes into account the ratio of the time that the robot spent as a moving robot over the total evaluation time: this is done in order to proportionate the distance between expected utility and

[3]a video of a simulation showing the amorph pattern is provided at https://vimeo.com/101705747

obtained utility in those robots that acted with supporting roles (pheromone or node robots). A detailed formulation of $r_{i,j}$ can be found in Eq. 2.

During the bootstrapping phase, each individual coordination pattern is tested in both arenas over a 50000 time-step period. At the end of each period, each robot logs its status and the amount of food it collected. These results were then averaged to build an initial set of 10 seed antibodies. Complexity is estimated during an additional phase in which the robots apply a diffusive algorithm and record the number of time they traverse between the nest and food area within a fixed interval. This is averaged to give a complexity value in which low values indicate complex arenas.

Results indicate different correlations between performance and arena-complexity for each coordination pattern, reinforcing the need for run-time adaptation. We also observe that run-time adaptation is necessary to enable the swarm to deal with local issues as they arise. For example, poor performance often results if stationary robots (e.g. pheromone robots from the rACO pattern) are placed too close to walls, or too distant from each other. This disrupts the ability of the swarm to compute accurate trajectories. Run-time adaptation should enable components to switch coordination pattern to circumvent this. Clearly some simple situations as just described are predictable and strategies to address these factors could be pre-coded. However, in any real, dynamic environment, it would not be possible to envisage all situations that may arise. A run-time adaptation strategy that can both detect issues and facilitate appropriate reconfiguration is therefore preferable.

4.1.5. Runtime Phase: Mechanism of adaptation. After $eval$ timsteps, each robot evaluates its own performance u and compares this to the expected utility u_E indicated in the currently active antibody.

Affinity values r (between antibody i and j) are calculated as follows:

$$\begin{aligned} r_{i,j} &= \omega|obtUtilities - expUtilities| + \frac{K_0}{|dAc - abAc|} \\ \omega &= \begin{cases} status & \text{if } status \text{ is} \leq 1 \\ 1 - \frac{status}{evalTime} & \text{if } status > 1 \end{cases} \end{aligned} \tag{2}$$

The difference between obtained utilities (*obtUtilities*) and expected utilities (*expUtilities*) is weighted according to the $status$ variables (see Section 4.1.4). *obtUtilities* simply counts how many food units have been collected during the evaluation interval; *expUtilities* is the value expected according to the previous bootstrapping phase. In addition, the affinity is adjusted according to the difference in the detected arena complexity (dAc) and the complexity value stored in the antibody ($abAc$), regulated by the constant K_0 (< 1). Thus, 2 represents the implementation specific version of Eq. 1(b). No other variations are required as there are no non-functional requirements in this application.

4.1.6. Runtime Phase: Antibodies mutation and knowledge sharing. In this case study, a mutation operator is not implemented, therefore each robot operates with the same set of antibodies produced during the bootstrapping phases. New antibodies are not introduced during runtime, nor are any antibodies removed or changed. Thus, only the concentrations vary during each experiment. However, new *global* coordination patterns emerge due to subsets of the swarm selecting different coordination patterns: Figure 4 clearly illustrates this in the two arenas.

As discussed in Section 3.2.3, *sharing* information between components enables one component to learn from another's experiences. Sharing is implemented here by a method in which each robot broadcasts the concentration of its m most concentrated

antibodies through infra-red communication actuators. In this case m is limited to 4 due to limitations on broadcast ability of the sensor. Receiving robots average the received concentrations with their own concentrations.

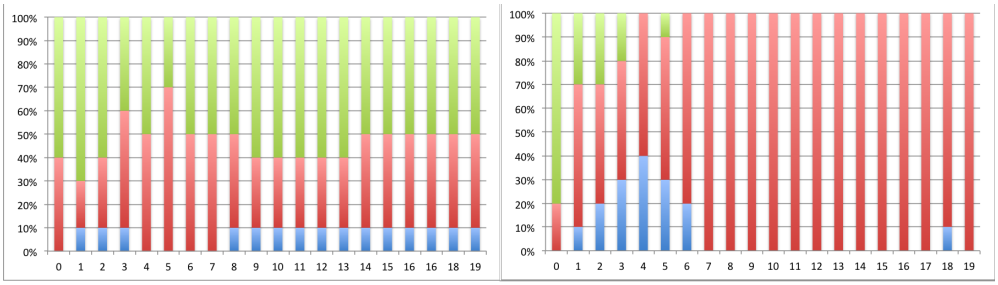

Fig. 4. Each graph represents a different arena and each colour represents a different coordination pattern as chosen by the shown percentage of robots during 20 simulated time intervals. Green is amorph, red is rACO and blue the blind diffusion. In the left arena, the swarm does not converge towards a shared decision. The opposite situation is shown on the right where tima a common decision is reached at time 6.

.

4.1.7. Results and findings. Detailed experiments and results are given in [Capodieci et al. 2013]. Here we highlight that introducing the ability for the swarm to exhibit self-expression through a specific instantiation of the generic framework described outperforms all of the individual coordination patterns. Results shows that in both arenas, robots learn over time, i.e. the amount of food collected during each evaluation period increases over the course of the experiments. We observe two patterns. In the simple arena, the algorithm leads to subsets of the swarm selecting different coordination patterns — the swarm reaches a consensus on this composition after some period of time. Considered from a global perspective, this can be viewed as the emergence of a new coordination pattern, formed as a composition of known strategies. On the other hand, in the more complex arena, the swarm reaches a common consensus to use a single pattern in order to optimise the task. This is shown in Figure 4.

This case-study shows that the generic framework was easily customised to an application in which the goal was to optimise task performance at run-time in a given environment. The results show that the system is capable of learning over time and adapting to different environments without intervention. Through the antibody matching mechanism, each component of the swarm is able to detect if task performance is deteriorating over time, via the decrease in antibody concentration level. This leads to a new coordination pattern being selected during the run time execution of the task, which in turn optimises performance.

In the next case-study, we extend the complexity in order to consider a system in which there are both functional and non-functional requirements.

4.2. Evolution in virtual creatures

In this case study we consider a scenario in which a set of independent components are aggregated into a virtual creature. Each component moves independently, hence there is no central control — movement of the creature as a whole emerges as a result of the independent movements of each component. There is a functional requirement for the aggregated swarm to maximise the distance it moves, and an additional non-functional requirement to minimise the energy consumed during movement. As before, we first

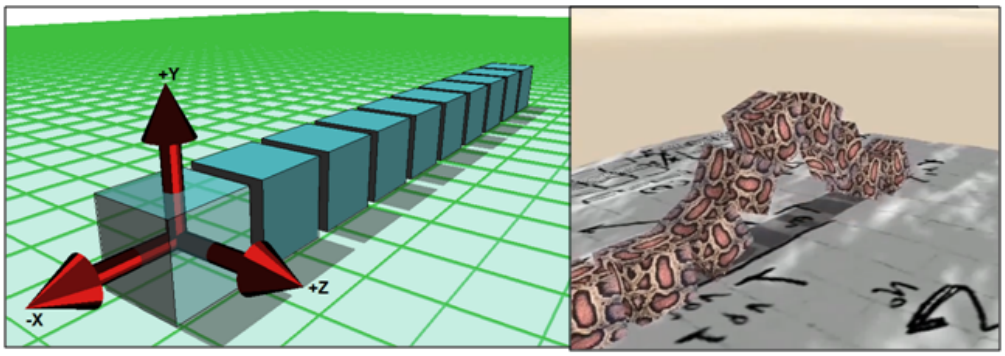

Fig. 5. (Left) a basic representation of the creature, showing morphology and movement capabilities. (Right) the creature is shown moving in its rendered simulated environment.

.

describe the scenario involved to then apply our design guidelines to enable run-time self-expression.

4.2.1. Related Work on Virtual Creatures. The vibrant field of Artificial life uses computer simulation to investigate evolution of behavioural and cognitive mechanisms in virtual creatures [Sims 1994], potentially leading to advances in both biology (e.g: [Palyanov et al. 2012]) and robotics (e.g: [Černỳ and Kubalík 2013]). We restrict our review to work related to understanding the evolution of movement strategies that might ultimately be applied to the robotic field. A significant volume of work exists in the evolutionary computing literature, summarised by [Prez-Moneo Surez and Rossi 2013] in relation to movement of limbless creatures. Typically, evolution evolves centralised controllers in which performance is evaluated in terms of the evolved trajectory and ability to avoid obstacles but does not account for energy consumption of the movement, a relevant factor if the motion is to be transferred to real robots. Moreover, our approach (detailed in Section 3) uses a set of pre-coded movement strategies as baseline behaviours to evolve.

4.2.2. Creature description. The virtual creature distributed controller we implemented, is a limbless animal composed of ten independent units, shaped as perfect cubes (see Figure 5). These cubes are connected through a series of universal joints, thus giving them a certain degree of freedom for rotation. The creature as a whole shares a biological clock — a periodic signal in the form of a square wave in which we identify a positive and a negative phase. The creature is inserted in a simulated three dimensional environment in which collisions, friction and gravitational forces are present. Starting from a restricted set of pre-coded movement strategies, the objective of the creature is to discover new locomotion patterns that enable it to maximise movement and minimise energy (calculated as the sum of the energy consumption of each of the individual units).

During each clock phase, each unit independently decides whether to apply a force from the centre of its mass with a specific magnitude and direction as described in the vectors visible in Figure 5. The direction of each force is always relative to the orientation of the unit. Movement of the creature results from physical interactions between moving units. Not all units are required to apply a force (and therefore consume energy) for movement to occur: push or pulling behaviour of a unit can result from a force applied to a neighbouring unit. The energy spent by a single unit during

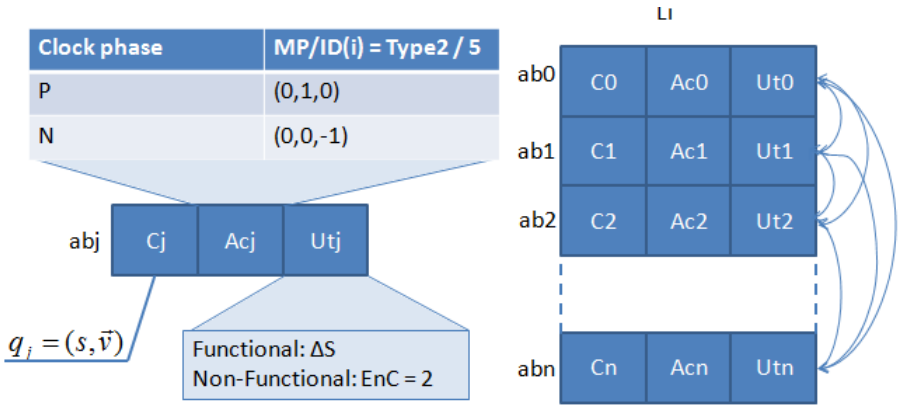

Fig. 6. A single antibody with example values (left) and a collection of interconnected antibodies inside a single lymph-node (right). In the single antibody, the condition field represents the unit orientation and it is expressed in quaternions. The action field, indicates magnitude and direction of the force for each clock phase (indicated as P for positive and N for negative). The expected utility field is indicated as ΔS for the expected distance to travel during a time interval, and the energy consumption required by the selected action.

a single time interval is calculated by summing the magnitudes of the applied force in both the positive and negative phase of the clock. The total energy consumed during a time interval is trivially calculated by summing the energy consumption experienced by each constituent unit during each phase of the clock.

4.2.3. Model. The modelling phase follows the same pattern as in the previous case study. The independent constituent units that shape the creature are modelled as lymph-nodes: each cube represents a lymph-node able to host a set of interconnected antibodies. As in the previous case study, each antibody is formulated as a tuple composed of three fields: a *condition* field, an *action* field and an *expected utility* field (see Figure 6).

4.2.4. Coordination Patterns, functional and non-functional utilities and conditions. Five precoded movement patterns are used as initial coordination patterns. A movement pattern is a description of how the whole creature moves and it is obtained by indicating the magnitude and direction of the forces to be applied to each of its constituent units during each phase of the clock. These movement patterns (which are indicated with the names *Type1* to *Type5* and can be seen at https://vimeo.com/89119516) are loosely based on how limbless creatures move in space and the single individual units iterate the application of such forces as described in a data structure that corresponds to the action field of the antibody tuple during every time interval. Each initial pattern has an associated energy consumption that is fixed through the duration of an experiment. Moreover, the initial patterns do not enable obstacle avoidance or adaptation to external perturbations that cause unexpected rotations of the unit(s) and thus provide a baseline for evaluating whether adaptive behaviours can emerge. Exactly as in previous case studies, different coordination patterns result in different outcomes both in terms of non-functional requirements (the total energy consumption from the point of view of the whole creature) and in terms of functional requirements (calculated as the space distance the whole creature managed to travel during a time interval). With respect to distance travelled, performance varies according to the orientation of the constituent units (the condition field, represented as quaternions), therefore any per-

$$(a)\ nfReq = K_D \frac{d(q, q')}{1 + EnC_i}$$
$$(b)\ r_{i,j} = \frac{T_{ni} + T_{pj}}{T_{i,j}}(1 + |\Delta S_o - \Delta S_{exp}|) \tag{3}$$

turbation that causes the units to rotate from their original position can drastically change its performance and even prevent the creature from moving. .

4.2.5. Bootstrapping Phase. Testing the five pre-coded movement patterns individually in the pre-experimental phase allows each unit to build an initial repertoire of antibodies, which are then used as the starting point for the on-line dynamic phase of the framework. In this phase, the different initial conditions are presented to the constituent elements of the creature by applying random disturbances to the whole creature: the physics engine used in the simulations allowed us to pick, throw, twist etc... the creature in order to provide sudden changes in the orientation for its constituent units. The simulation environment is a custom made 3D environment coded in java (JOGL[4] for OpenGL bindings and jinngine[5] for the physics engine).

4.2.6. Runtime Phase. The dynamics are executed exactly as formulated in the previous Section 3.2. For this study, $r_{i,j}$ is defined as in Eq. 3, which provides a specialised instantiation of the general Eq. 1.

Affinities and concentration values are calculated periodically every T_{eval}. Eq. 3 specifies the non-functional requirements for the applicaton (i.e the third term in Eq. 1. K_D is simply a constant value used to regulated the impact of non-functional requirements and observed conditions over the calculation of the concentration variation for the single antibody. EnC_i is the value of consumed energy by lymph-node i for the current evaluation time. In terms of specifying affinity, the first term of Eq. 3 differs from the one in Eq. 1 by a term that depends on the movement of the creature. More specifically, we define as Δs the Euclidean distance between the starting position of the unit and its position at the end of a single time interval, in order to calculate the difference between the movement obtained in the current evaluation interval (Δs_o) and the movement expected from the bootstrapping phase and stored in the utility field of the selected antibody (Δs_e). Therefore, the affinity term $r_{i,j}$ once again depends on the functional requirement we specified for this task.

The similarities with the previous case studies are evident, as the concentration updates in this scenario are also modelled as detailed in Section 3. However there are significant differences in this case study with respect to information sharing and mutation.

In this case study, a *mutation* operator is used to generate new solutions. Two different mutation operators are applied whenever stagnation occurs in the network (see Section 3.2.2): if more than one antibody reports the same maximum level of concentration, these antibodies are combined (i.e. their tuple values are mathematically averaged) to create a new antibody. A second mutation operator is activated to create new antibodies from a single antibody whenever the first mutation operator fails to deliver expected results (described in detail in [Capodieci et al. 2014b].

Finally, a sharing procedure is implemented to enable sharing of experiences. This differs from the previous study given that fact that in this case the swarm has fixed topology. We consider additional links that connect every antibody i in a component l to

[4]http://jogamp.org/jogl/www/
[5]https://code.google.com/p/jinngine/

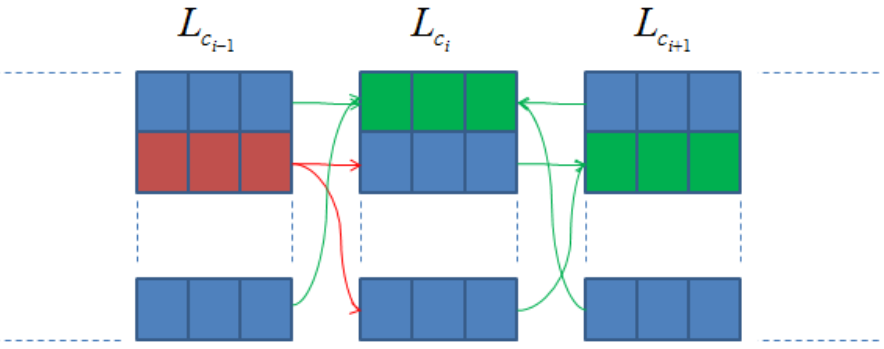

Fig. 7. The external idiotypic network used for experience sharing among neighbouring lymphnodes. Red antibodies provide negative feedback, green ones provide positive feedback. Arrows symbolize the suppression and stimulation signals in the form of varying inter-antibodies affinities.

every antibody in components $(l+1)$ and $(l-1)$. If the antibody selected in component l exceeds its expected utility, then all other antibodies in neighbouring networks provide positive feedback. Vice-versa, in that utility is lower than predicted, the antibody will receive negative feedback from all connected antibodies in both its external and internal network. (see Figure 7). In essence, this results in the algorithm considering the total concentration update for the single antibody as a weighted composition of the effects of both the internal and external immune networks.

4.2.7. Results. Results show that the approach is able to adapt the initial movements into new strategies, which for example enable it to avoid simple obstacles and additionally minimise energy consumption, compared to the initial strategies. For a more detailed analysis of these assertions, the reader in once again referred to [Capodieci et al. 2014b]. As in the previous study, the ability to exhibit self-expression provides improved performance with respect to each individual coordination pattern.

In particular, when the creature becomes stuck, the reduction in concentration of the selected antibodiy activates the adaptation mechanism, i.e. selection of a new antibody by each unit. This results in a new coordination pattern emerging at the global level, i.e. a new movement mechanism for the creatures as whole. Mutation provides an additional mechanism for discovering new movement strategies. In addition, the inclusion of the non-functional requirement relating to energy consumption always results in selection of the least energy consuming action.

4.3. Morphogenetic Engineering

The third and last case study deals with a more abstract system known as *Swarm Chemistry*, chosen for its potential to study the concept of self-expression in a complex collective system in which the objective is to discover completely new architectures.

Swarm Chemistry was introduced by [Sayama 2007] through his work in the field of morpho-genetic engineering [Doursat et al. 2012]. This relatively new domain aims to develop methods by which programmable self-organisation can be introduced into engineered (and therefore architected) systems, borrowing ideas from natural systems in which organisation emerges from systems in which there is no initial organisation or architecture. Specifically, swarm chemistry concerns systems of particles that coexist in the same space but self-organize in interesting oscillatory and shape-formation behaviours. Each particle moves according to a flocking algorithm that is characterised by kinetic parameters assigned to each of the swarm constituent individual (i.e. sens-

ing radius R, normal speed V_n, maximum speed V_m, strength of cohesive force c_1, strength of aligning force c_2, strength of separating force c_3, probability of random steering c_4 and tendency for self-propulsion c_5).

When a single tuple of kinetic parameters is assigned to a whole swarm, the homogeneous swarm is able to show dispersal, coherent linear and even oscillatory motion patterns, depending on the parameter set chosen. Much more interesting and yet unpredictable behaviours arise when the swarm is heterogeneous, i.e. when different swarms characterized by different kinetic parameters are confined within the same space and they are not aware of their differences. This gives rise to the name *swarm chemistry*, as the process of mixing different *recipes* (as different values of assigned kinetic parameters) together in order to create new robust structures or other notable behaviours. A single recipe r that characterizes a heterogeneous swarm is written in the form:

Recipe r:

$$P_1 * (R_0, V_{n_0}, V_{m_0}, c_{1_0}, c_{2_0}, c_{3_0}, c_{4_0}, c_{5_0})$$
$$P_2 * (R_1, V_{n_1}, V_{m_1}, c_{1_1}, c_{2_1}, c_{3_1}, c_{4_1}, c_{5_1})$$
$$\vdots$$
$$P_M * (R_M, V_{n_M}, V_{m_M}, c_{1_M}, c_{2_M}, c_{3_M}, c_{4_M}, c_{5_M})$$

Recipe r is composed of M different *types*, being P_i, $i \in [1, ..., M]$, the number of individuals of the swarm that are currently characterized by the i^{th} type of recipe. The reader is directed to the swarm chemistry website to inspect samples of heterogeneous swarms with mixed recipes (http://bingweb.binghamton.edu/∼sayama/SwarmChemistry/).

In this specific case study, the purpose of applying our framework is two-fold; firstly, to show that the system is able to evolve new recipes (architectures) over time, and secondly, to be able to autonomously detect whether the new architecture has been formed without visual inspection. This approach enables both a deeper understanding of the emergent properties of self-organising systems and the role of self-expression in such a system.

4.3.1. Model. According to the design methodology detailed in Section 3 and utilized in the previous applications, a single self-propelled particle within a swarm is associated to a lymph-node. The method by which antibodies are modelled inside the single lymph-node is shown in Figure 8. Note in this specific case, there is no *condition* field in the antibody tuple as there is no functional requirement. The *action* field (Ac_j) is a formal description of a multi-type recipe. An individual action is selected from the antibody through a process of stochastic differentiation: given a recipe composed of N types, then the i^{th} tuple of kinetic parameters will have a probability p_i of being selected that is dependent on the ratio between the population of the i^{th} type of the recipe and the total population indicated by the recipe stored in the antibody.

The *expected utility* field (Ut_j) is a real value that is calculated according to a specially designed fitness function. The purpose of this fitness function is to balance the number of similar individuals (i.e. those particles that share the same kinetic parameters) against the number of individuals with different kinetic parameters observable in the same radius, at the same time avoiding concentrating similar or diverse particles inside a restricted neighbouring radius. H_o and H_e represent the indexes of local *homogeneity* and local *heterogeneity* (as concentration of similar and diverse particles inside the particle's observation radius). Using a pseudo-Gaussian formula detailed in Eq. 4, we assign higher fitness values to those particles in which the values of both indices are located close to the midpoint of their possible value range. In Eq. 4, N_R is

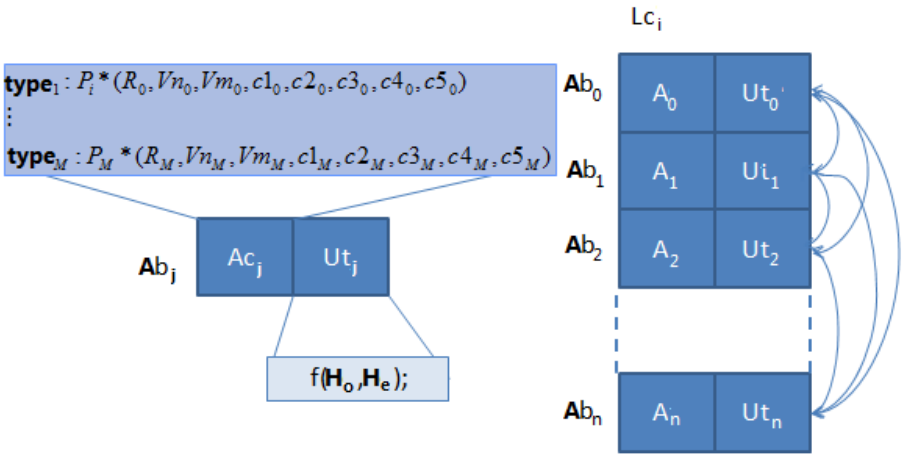

Fig. 8. Antibody modelling in the swarm chemistry case study. The expected utility field is a function f; the action field is a formal description of a multi-type recipe.

the total number of neighbour individuals, K_0 is a constant that is used to move the maximum value of the function according to a specified value of local heterogeneity index.

$$fitness_{(H_o,H_e)} = exp\left(-\left[\left(\frac{H_o - \frac{N_R}{2}}{N_R}\right)^2 + \left(\frac{H_e - \frac{K_0}{2}}{K_0}\right)^2\right]\right) \quad (4)$$

4.3.2. Bootstrapping Phase. The initial repertoire is once again created through a bootstrapping phase. Here, a collection of seventeen recipes taken from the official swarm chemistry website were used as the initial coordination patterns and tested individually. At the end of a designated time interval, the obtained fitness value is stored in the expected utility field of the virtual antibody. As in the swarm-robotics study (but differently from the virtual creature) the initial repertoire of antibodies loaded inside each particle is identical for each lymph-node.

4.3.3. Runtime phase. The on-line dynamics follows the steps detailed in Section 3.2, with the exception of the initialisation phase. During the first time interval, each component of the swarm makes a random selection from its seventeen antibodies. A stochastic differentiation procedure is then applied to the selected recipe stored in the action field of the selected antibody, i.e. each unit selects a tuple i of kinetic parameters from a randomly selected recipe r with a probability $p_{s_i} = P_i/(P_1 + ... + P_M)$, where P_x following the same notation for recipes introduced earlier in this section.

This results in the swarm converging to new formations and behaviours from the initial chaotic movement[6], through the calculation of new inter-antibodies affinities using Eq. 1(b).

A stagnation condition (defined in the same way as the previous case study) triggers mutation: in this case this is implemented as the recombination of the highest concentrated antibodies, creating new recipes by averaging the kinetic parameters of the

[6]see https://vimeo.com/92732849)

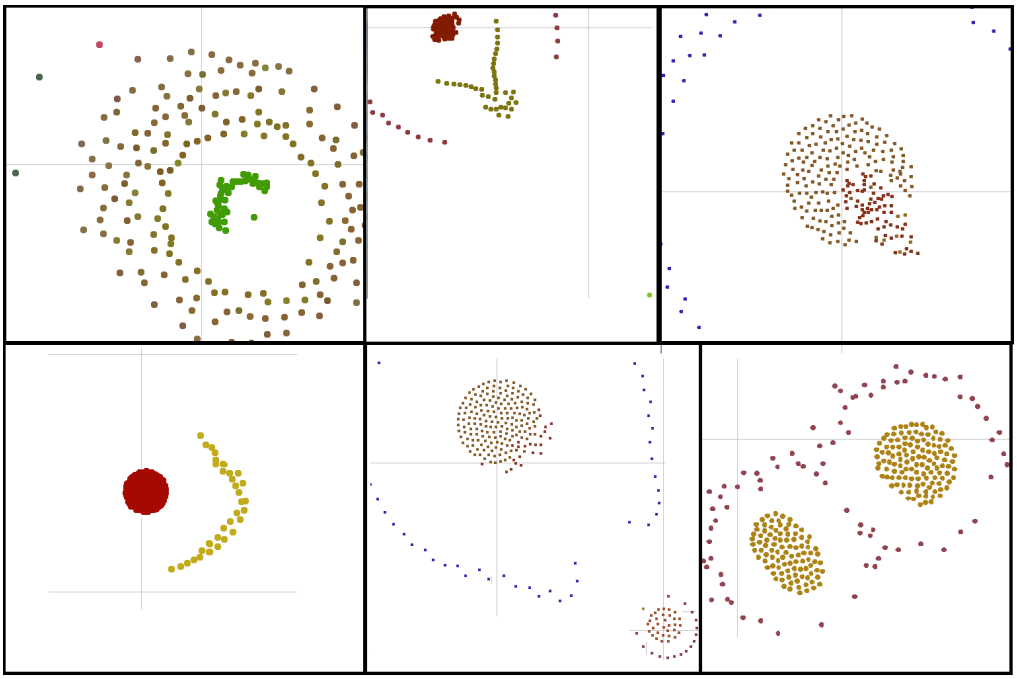

Fig. 9. Example of novel configurations that emerged when applying the framework to the swarm chemistry case study

involved recipes. The *sharing* of the experiences in this case study is implicit: similar particles tend to stay close and share the same observation radius, and thus they are likely to record very similar performances.

An additional step in this study enables the swarm to identify novel emergent recipes. The original seventeen recipes form a *self*-set: as new recipes are evolved, they are added to this set if they are deemed to be sufficiently dissimilar to any recipe currently in self. Dissimilarity is defined in terms of a distance metric as follows:

Two dimensions are defined: the first combines the values of the kinetic parameters of the types composing the recipe in the form of a weighted average; the second dimension is a fragmentation index that takes into account how the extent to which the swarm is split among different types. The Euclidean distance between these two real values for each recipe provides an appropriate distance measure that captures appropriate features of a recipe. Some example recipes that the system autonomously labelled as new and added to the self-set are depicted in Figure 9.

4.3.4. Results and Findings. During each time interval, all recipes generated before the convergence criterion is reached are stored and compared to the *self-set* according to a *distance measure*. Figure 9 shows the unique new recipes that the swarm was able to discover in the completely autonomous manner we just described.

As in the previous case-studies, the section has described how the generic framework can be specified for a particular application. Very few modifications were required compared to two robotic studies. Of particular note here is that the algorithm was implemented on the top of an existing implementation of an artificial chemistry framework and involves mutations at the level of mixing pre-existing recipes and in the generation of new tuples of kinetic parameters to be assigned to particles via a newly introduced

genetic operator. In addition, a novel measure of distance among recipes enabled the system to autonomously detect novel recipes.

5. CONCLUSIONS AND FUTURE WORK

This paper has presented a conceptual framework consisting of a series of design principles and customisable algorithms that can be used within a collective system to enable self-expression. The conceptual framework encapsulates the key features proposed in [De Lemos et al. 2013] to be essential in designing a new generation of self-organising systems:

— *Representation*: the antibody structure defines information that can be sensed from the environment, alongside an expected utility of applying a particular action
— *Observations* of the system describing environmental conditions are matched against those stored in antibodies, which influences behaviour selection
— *Control*: this is realised via set of dynamic equations that vary the concentration of each antibody, capturing preference relationships between antibodies themselves and between antibodies and the environment
— *Identify*: possible solutions are identified in terms of antibodies displaying the highest concentrations. In addition mutation operators enable new potential solutions to be explored, thereby expanding the search space.
— *Enact adaptation*: selection of the antibody with the highest concentration determines the action that a particular component should apply

The generality of the framework has been examined through applying it to three different case-studies, each highlighting different properties, including its ability to adapt behaviour to optimise functional and non-functional requirements at runtime, as well as to discover novel architectures. Table 5 summarises how the framework was adapted to each of the case studies, highlighting both the commonalities and customisations made in each case.

Although the framework does not specify what either the number or type of coordination patterns should be included, in line with immunological principles outlined in [Cohen 2000b], it is preferable that selected patterns should share common elements such that one pattern acts as scaffolding for another. This is clearly demonstrated in the first case study in swarm robotics in which the three selected coordination patterns share commonalities as given in Table I. In practical terms, this has the desirable effect that a change in coordination pattern is less likely to result in a sharp drop in performance and less likely to lead to sub-swarms exhibiting antagonistic behaviours. However, the designer must take care that there are no obvious conflicts between assigned values in different coordination patterns (e.g. in colours chosen as indicators etc.).

Clearly, some elements are application dependent at the implementation stage. For example, the system engineer can and should tune the specific representation and related mutation operators according to the specific application and requirements. Another aspect that require specific tuning is setting an appropriate length of time to evaluate ensemble performance, before updating antibody concentrations. To be more specific regarding design choices in different case studies, in Table 5 we highlight the different modelling choices we operated in the three different applications we presented in the previous section.

5.1. Future Work

A number of fruitful avenues for research remain. Much of the existing literature in the AIS domain utilising idiotypic networks incorporates methods that enable the composition of a network to adapt over time, for example adding new antibodies and re-

Table II. Commonalities and differences in modelling choices for the three different case studies.

Case study	**Antibody Tuple**	**Mutation**	**Experience sharing**	**Initial antibody diversity**
Swarm Robotics	conditions, action, exp. utility	none	Antibodies diffused within communication range	low
Virtual Creature	condition, action, exp. utilities	antibody level	antibodies shared via inter-lymphnode (dual) immune network	high
Swarm chemistry	action, exp. utility	antibody level	implicit	low

moving weak or redundant ones. Exploring additional methods to generate novel antibodies will inevitably benefit the approach by improving exploration of the search space of potential solutions.

The utility metric drives the system towards fitter solutions, in terms of the functional and/or non-functional requirements of the system. However, recent literature in the optimisation field points towards the use of *novelty* as a metric to drive search rather than the use of an objective function [Lehman and Stanley 2011]. This might have particular benefits in applications such as the final swarm-chemistry case study in which we wish to discover completely novel solutions or even architectures, i.e. perform open-ended evolution without an explicit objective function. Investigation of more specific and effective mutation operators for evolving/generating new antibodies will also enhance of the ability of the system to discover novel behaviours.

Application to a wider variety of case studies will demonstrate further confidence. In particular, it will be important to evaluate the performance of the framework in applications in which there are very large ensembles and in which components can be dynamically added or removed, as in these situations it may not be possible to synchronise evaluation of the lymph-nodes, a factor that has been assumed in the current implementations. The impact of the initial state on the final outcome should also be investigated further; in particular this may have influences on the bootstrapping phase that are currently under-investigated. In addition to measuring fitness in terms of functional requirements, future effort should also be directed towards validation and verification of the resulting systems, for example through model checking of the systems. Finally, the effect of the modelling choices made the system designer during the early stages of the design process should be further investigated, particularly in relation to the choice of the set of coordination patterns. In the future, if we wish to apply the method to systems in which coordination patterns and task definitions are continuously added, it will be essential to ensure that conflict does not arise as a result of the self-expression mechanism selecting subsets of different patterns. By exploiting commonalities at the level of code among the implemented coordination patterns as previously discussed, this should be mitigated.

6. ACKNOWLEDGMENTS

The work is partially supported by the ASCENS project (EU FP7-FET, Contract No. 257414).

REFERENCES

Harold Abelson, Don Allen, Daniel Coore, Chris Hanson, George Homsy, Thomas F Knight Jr, Radhika Nagpal, Erik Rauch, Gerald Jay Sussman, and Ron Weiss. 2000. Amorphous computing. *Commun. ACM* 43, 5 (2000), 74–82.

Dhaminda B Abeywickrama, Nicola Bicocchi, and Franco Zambonelli. 2012. SOTA: Towards a general model for self-adaptive systems. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*. IEEE, IEEE, Piscataway, NJ, USA, 48–53.

Tomas Bures, Rocco De Nicola, Ilias Gerostathopoulos, Nicklas Hoch, Michal Kit, Nora Koch, Giacoma Valentina Monreale, Ugo Montanari, Rosario Pugliese, Nikola Serbedzija, and others. 2013. A life cycle for the development of autonomic systems: The e-mobility showcase. In *Self-Adaptation and Self-Organizing Systems Workshops (SASOW), 2013 IEEE 7th International Conference on*. IEEE, IEEE, Piscataway, NJ, USA, 71–76.

Giacomo Cabri and Nicola Capodieci. 2013. Runtime Change of Collaboration Patterns in Autonomic Systems: Motivations and Perspectives. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*. IEEE, IEEE, Piscataway, NJ, USA, 1038–1043.

Giacomo Cabri, Nicola Capodieci, Luca Cesari, Rocco De Nicola, Rosario Pugliese, Francesco Tiezzi, and Franco Zambonelli. 2014. Self-expression and dynamic attribute-based ensembles in SCEL. In *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*. Springer, 147–163.

Nicola Capodieci, Giacomo Cabri, and Franco Zambonelli. 2014. Modeling Self-Expression by holons. In *High Performance Computing & Simulation (HPCS), 2014 International Conference on*. IEEE, Piscataway, NJ, USA, 424–431.

Nicola Capodieci, Emma Hart, and Giacomo Cabri. 2013. An immune network approach for self-adaptive ensembles of autonomic components: a case study in swarm robotics. In *Advances in Artificial Life, ECAL*, Vol. 12. MIT press, Cambridge, MA, USA, 864–871.

Nicola Capodieci, Emma Hart, and Giacomo Cabri. 2014a. Artificial Immune System driven evolution in Swarm Chemistry. *Proceedings of The Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems SASO 2014* to appear, to appear (2014), to appear.

Nicola Capodieci, Emma Hart, and Giacomo Cabri. 2014b. Idiotypic networks for evolutionary controllers in virtual creatures.. In *Proceedings of The 14th International Conference on the Synthesis and Simulation of Living Systems ALIFE 2014*. MIT Press, Cambridge, MA, USA, 192–199.

Jan Černỳ and Jiří Kubalík. 2013. Co-evolutionary approach to design of robotic gait. In *Applications of Evolutionary Computation*. Springer, 550–559.

I.R. Cohen. 2000a. Discrimination and dialogue in the immune system. *Seminars in immunology* 12, 3 (2000), 215–219.

Irun R Cohen. 2000b. *Tending Adam's Garden: evolving the cognitive immune self*. Academic Press, Waltham, Massachusetts, USA.

Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, and others. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*. Springer, Berlin, DE, 1–32.

Rocco De Nicola, Michele Loreti, Rosario Pugliese, and Francesco Tiezzi. 2014. A formal approach to autonomic systems programming: the SCEL Language. *ACM Transactions on Autonomous and Adaptive Systems* 9, 2 (2014), 1–29.

Tom De Wolf and Tom Holvoet. 2007. Design patterns for decentralised coordination in self-organising emergent systems. In *Engineering Self-Organising Systems*. Springer, 28–49.

Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. 1996. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 26, 1 (1996), 29–41.

René Doursat, Hiroki Sayama, and Olivier Michel. 2012. *Morphogenetic engineering: toward programmable complex systems*. Springer, Berlin, DE.

Jose Luis Fernandez-Marquez, Giovanna Di Marzo Serugendo, Sara Montagna, Mirko Viroli, and Josep Lluis Arcos. 2013. Description and composition of bio-inspired design patterns: a complete overview. *Natural Computing* 12, 1 (2013), 43–67.

Dario Floreano and Claudio Mattiussi. 2008. *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press, Cambridge, MA, USA.

Alex Alves Freitas and Jonathan Timmis. 2007. Revisiting the foundations of artificial immune systems for data mining. *Evolutionary Computation, IEEE Transactions on* 11, 4 (2007), 521–540.

Emma Hart and Jon Timmis. 2008. Application areas of AIS: The past, the present and the future. *Applied soft computing* 8, 1 (2008), 191–201.

Akio Ishiguro, R Watanabe, and Yoshiki Uchikawa. 1995. An immunological approach to dynamic behavior control for autonomous mobile robots. In *Intelligent Robots and Systems 95.'Human Robot Interaction*

and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on, Vol. 1. IEEE, IEEE, Piscataway, NJ, USA, 495–500.

M.K. Jerne. 1974. Towards a network theory of the immune system. Annals of Immunology. *Annals of immunology, 2d ed. Cambridge* 125C (1974), 373–389.

Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.

Carles G Orosz. 2001. An introduction to immuno-ecology and immuno-informatics. In *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*. Reading, Mass, Addison-Wesley, Boston, ME, USA, 125–150.

Andrey Palyanov, Sergey Khayrulin, Stephen D Larson, and Alexander Dibert. 2012. Towards a virtual C. elegans: A framework for simulation and visualization of the neuromuscular system in a 3D physical environment. *In silico biology* 11, 3 (2012), 137–147.

Carlo Pinciroli, Vito Trianni, Rehan OǴrady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, and others. 2012. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence* 6, 4 (2012), 271–295.

Mariachiara Puviani, Giacomo Cabri, and Letizia Leonardi. 2014. Enabling Self-expression: the Use of Roles to Dynamically Change Adaptation Patterns. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2014 IEEE Eighth International Conference on*. IEEE, 14–19.

Mariachiara Puviani, Carlo Pinciroli, Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli. 2013. Is self-expression useful? Evaluation by a case study. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*. IEEE, IEEE, Piscataway, NJ, USA, 62–67.

Dmaso Prez-Moneo Surez and Claudio Rossi. 2013. A Comparison between Different Encoding Strategies for Snake-Like Robot Controllers. In *Applications of Evolutionary Computation*. Lecture Notes in Computer Science, Vol. 7835. Springer Berlin Heidelberg.

Hiroki Sayama. 2007. Decentralized control and interactive design methods for large-scale heterogeneous self-organizing swarms. In *Advances in Artificial Life*. Springer, Berlin, DE, 675–684.

Karl Sims. 1994. Evolving 3D morphology and behavior by competition. *Artificial life* 1, 4 (1994), 353–372.

A.M. Whitbrook, U. Aickelin, and J.M. Garibaldi. 2010a. Real-world Transfer of Evolved Artificial Immune System Behaviours between Small and Large Scale Robotic Platform. *Evolutionary Intelligence* 3(3) (2010), 123–136.

A.M. Whitbrook, U. Aickelin, and J.M. Garibaldi. 2010b. Two-timescale learning using idiotypic behaviour mediation for a navigating mobile robot. *Appl. SoftComput* 10, 3 (2010), 876–887.

Franco Zambonelli, Nicola Bicocchi, Giacomo Cabri, Letizia Leonardi, and Mariachiara Puviani. 2011. On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2011 Fifth IEEE Conference on*. IEEE, IEEE, Piscataway, NJ, USA, 108–113.